
# 살려조

## 개인 홈트레이닝 자세 인식 및 교정

조원: 이인혁, 최원빈, 한민규

# Contents



# 조원별 역할

## 최원빈

Main : 팀장
Sub : 개발
회의록 및 작업일지 작성


## 이인혁

Main : 개발
Sub : 부팀장

## 한민규

Main : 기획
Sub : 시간관리

# 01 개요

## 추진 배경

- **비용과 시간 부담:** 최근 헬스와 웰빙에 대한 관심이 높아졌지만, 헬스장 등지에 시간과 비용을 지불하며 장기간 다니기에는 부담스러운 사람들이 많아 졌다.

- **개인 홈 트레이닝의 부상 우려:** 개인 홈 트레이닝의 인기 상승과 함께 부정확한 자세로 인한 부상 우려도 함께 증가하고 있다.

# 01 개요

## 구현 목표 및 내용

- **자세 인식 및 교정:** 영상 프레임 분석을 통해 개인의 운동 자세를 실시간으로 인식하고, 부정확한 자세를 교정하는 기능을 구현한다.

- **맞춤형 트레이닝 계획:** 사용자가 자신의 목표 및 체력 수준을 고려하여 운동 횟수, 휴식시간을 직접 조정한다.

# 01 개요

## 기대 효과

- **비용 절감 및 시간 절약:** 헬스장 회원 비용을 절감을 통해 경제적 부담을 낮추고 개인 여가 시간에 효율적으로 운동 할 수 있다.
- **부상 예방 및 정확한 자세교정:** 자세 인식 및 교정을 통해 부상을 방지하고 정확한 운동 자세로 트레이닝을 진행함으로써 건강한 운동 습관을 형성 할 수 있다.
- **맞춤형 트레이닝으로 효과적 성과 달성:** 사용자가 자신의 목표와 체력 수준을 고려하여 트레이닝을 진행하기에 운동 목표에 더욱 효과적으로 도달할 수 있다.

# 02 수행 절차 및 방법

## 개발 환경 및 장비

Python

Spyder

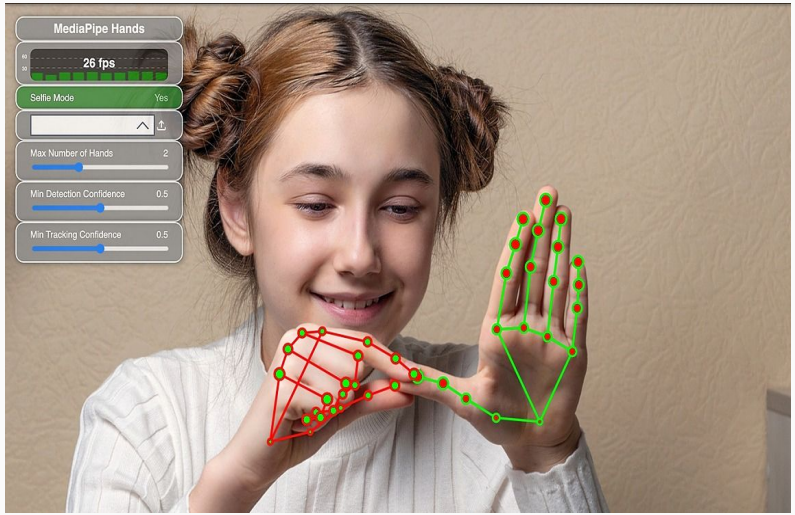

Camera

# 02 수행 절차 및 방법

## MediaPipe 소개

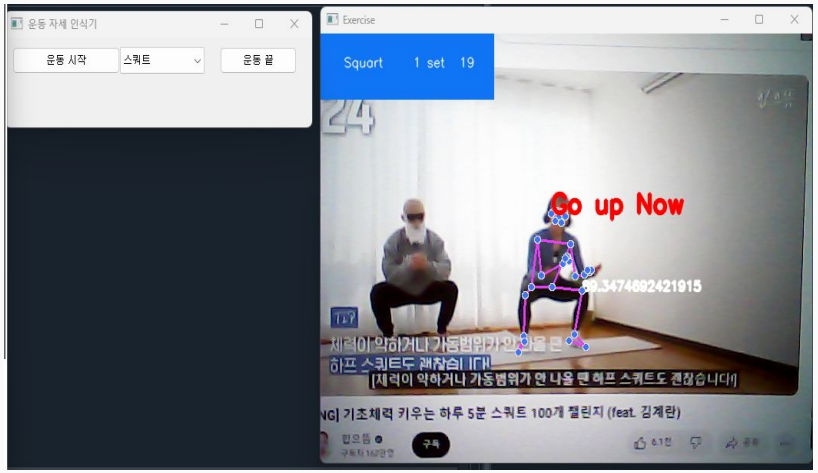
영상에서 얼굴 검출, 얼굴 그물망 검출, 손 검출, 자세 추정 등을 수행하는 **오픈 소스 라이브러리**

- **다양한 작업 지원:** 객체 감지, 자세 추적, 얼굴 및 손 인식 등
- **실시간 응용:** 높은 성능과 정확성으로 실시간 움직임을 추적 및 분석
- **다양한 플랫폼 호환:** 모바일 디바이스부터 데스크탑 까지 다양한 플랫 폼에서 사용가능



# 02 수행 절차 및 방법

## 기능 계획




<기능>

**텍스트 창 : 시작, 모드 선택**

- 운동 시작, 운동 끝 선택
- 팔굽혀펴기, 스쿼트 선택

**Window 창 : 실시간 운동 정보 표시**

- 실시간 운동 영상
- 운동 횟수 셋트 표기
- 운동 자세 랜드마크 표시
- 잘못된 자세 교정 문구 표시

**TTS : 잘못된 자세 감지 시 교정 음성 출력**

- “더 내려 가세요”, “이제 올라 가세요”등

# 02 수행 절차 및 방법

## 프로젝트 기간 별 계획

**2/14 ~ 15 :** MediaPipe 활용 예제 프로그램 구동 (o)

**2/16 :** 팔굽혀펴기 인식 프로그램 구동 (o)

**2/17 ~ 18 :** PyQt를 이용한 텍스트 창 + 스쿼트 인식 프로그램 구동 (o)

**2/19 ~ 21 :** 잘못된 자세 교정 프로그램 구동 (o)

**2/20 ~ 21 :** PPT 작성 및 프로젝트 마무리 (o)

# 03 프로젝트 과정

## 핵심 기술 상세 (랜드마크 좌표값 검출 및 출력)

```python
# 푸쉬업
if exercise_type==0:
    count_types = "PushUp"
    # 랜드마크 추출
    try:
        landmarks = results.pose_landmarks.landmark

        # 어깨, 팔꿈치, 팔목 값 저장
        left_shoulder = [landmarks[self.mp_pose.PoseLandmark.LEFT_SHOULDER.value]
                         .x,landmarks[self.mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
        left_elbow = [landmarks[self.mp_pose.PoseLandmark.LEFT_ELBOW.value]
                      .x,landmarks[self.mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
        left_wrist = [landmarks[self.mp_pose.PoseLandmark.LEFT_WRIST.value]
                      .x,landmarks[self.mp_pose.PoseLandmark.LEFT_WRIST.value].y]

        self.angle1 = calculate_angle(left_shoulder, left_elbow, left_wrist)

        # 계산된 각도를 팔꿈치 위치에 표시
        cv2.putText(image, str(self.angle1),
                       tuple(np.multiply(left_elbow, [640, 480]).astype(int)),
                       cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

    except:
        pass
```

- 미디어 파이프를 통해 감지된 포즈의 **랜드마크 정보**를 저장
- 왼쪽 어깨, 왼쪽 팔꿈치, 왼쪽 손목의 **좌표를 저장**
- 계산된 각도를 window창에 표시하여 사용자에게 **시각적 피드백** 제공

# 03 프로젝트 과정

## 핵심 기술 상세(관절 각도 계산)

```
# 각 값을 받아 넘파이 배열로 변형
a = np.array(a) # 첫번째
b = np.array(b) # 두번째
c = np.array(c) # 세번째

# 라디안을 계산하고 실제 각도로 변경한다.
radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
angle = np.abs(radians*180.0/np.pi)

# 180도가 넘으면 360에서 뺀 값을 계산한다.
if angle >180.0:
    angle = 360-angle

# 각도를 리턴한다.
return angle
```

- 관절 좌표(x,y) 값을 **넘파이 배열**로 저장
- **np.arctan2 함수**를 사용하여 세 점 간의 각도를 **라디안으로 계산** 후 이를 실제 **각도로 변환**

세 점 (a, b, c) 간의 각도를 라디안으로 계산하는 공식은 다음과 같습니다:

$$\text{angle} = \arctan 2(c_y - b_y, c_x - b_x) - \arctan 2(a_y - b_y, a_x - b_x)$$

여기서:

- $(a_x, a_y), (b_x, b_y), (c_x, c_y)$는 각각 점 A, B, C의 좌표입니다.

# 03 프로젝트 과정

## 핵심 기술 상세(관절 각도 계산)

```
# 각 값을 받아 넘파이 배열로 변형
a = np.array(a) # 첫번째
b = np.array(b) # 두번째
c = np.array(c) # 세번째

# 라디안을 계산하고 실제 각도로 변경한다.
radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
angle = np.abs(radians*180.0/np.pi)

# 180도가 넘으면 360에서 뺀 값을 계산한다.
if angle >180.0:
    angle = 360-angle

# 각도를 리턴한다.
return angle
```

- 각도가 180도를 넘으면 이를 보정

이후, 계산된 각도를 라디안에서 도로 변환하기 위해서는 다음과 같은 수식을 사용합니다.

$$\text{angle} = \text{angle} \times \frac{180.0}{\pi}$$

이 수식은 라디안에서 도로 변환하는 공식으로, 파이($\pi$)를 사용하여 라디안 값을 도로 변환합니다.

- 최종적으로 계산된 **각도를 반환**

# 03 프로젝트 과정

## 핵심 기술 상세 (팔굽혀펴기 자세 감지 및 교정)

```
# 운동횟수
if self.stage_pushup == "up":
    if self.angle1 < 70:
        cv2.putText(image, 'Go up Now',(300,200), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 0, 255), 3, cv2.LINE_AA)
        threading.Thread(target=self.play_sound, args=(file_name2,)).start()
        self.stage_pushup = "down"
    if 80 <= self.angle1 < 140:
        cv2.putText(image, 'Lower your posture',(300,200), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 0, 255), 3, cv2.LINE_AA)
        if not self.sound_played_pushup:
            threading.Thread(target=self.play_sound, args=(file_name1,)).start()
            self.sound_played_pushup = True
elif self.stage_pushup == "down":
    if self.angle1 > 160:
        self.stage_pushup = "up"
        self.sound_played_pushup = False
        self.counter += 1
```

사용자가 올라간 상태 일때:

- 팔꿈치 각도가 70도 미만이라면 **“Go up Now”메시지를 표시**하고 **음성 안내를 실행**
- 팔꿈치 각도가 80~140도 라면 **“Lower your posture” 메시지를 표시**하고, **음성 안내를 실행**

사용자가 내려간 상태 일때:

- 팔꿈치 각도가 160도를 초과하면 다시 **올라가는 동작으로 감지** 및 **횟수 카운트**

# 03 프로젝트 과정

## 핵심 기술 상세 (스쿼트 자세 감지 및 교정)

```
# 운동횟수
if self.stage_squart == "up":
    if self.angle2 < 90:
       cv2.putText(image, 'Go up Now',(300,200), cv2.FONT_HERSHEY_SIMPLEX,
                           1, (0, 0, 255), 3, cv2.LINE_AA)
       threading.Thread(target=self.play_sound, args=(file_name2,)).start()
       self.stage_squart = "down"
    if 95 <= self.angle2 < 150:
        cv2.putText(image, 'Lower your posture',(300,200), cv2.FONT_HERSHEY_SIMPLEX,
                           1, (0, 0, 255), 3, cv2.LINE_AA)
        if not self.sound_played_squart:
            threading.Thread(target=self.play_sound, args=(file_name1,)).start()
            self.sound_played_squart = True
elif self.stage_squart == "down":
    if self.angle2 >160:
        self.stage_squart = "up"
        self.sound_played_squart = False
        self.counter += 1
```

사용자가 올라간 상태 일때:

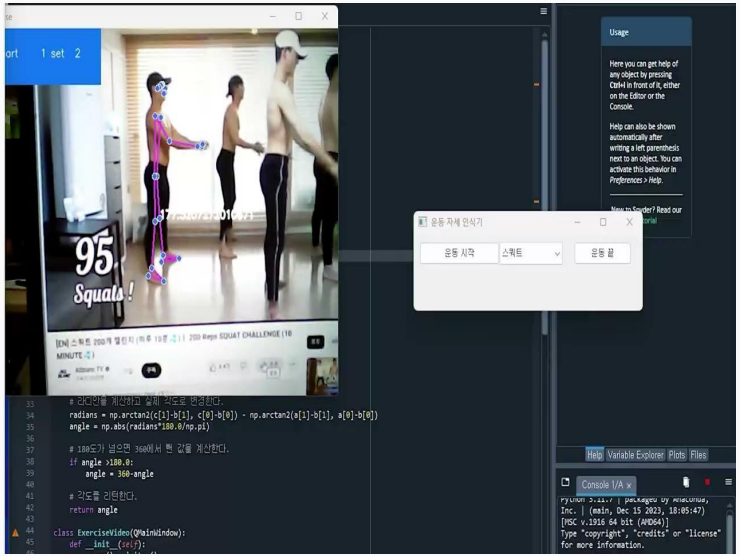
- 무릎 각도가 90도 미만이라면 **“Go up Now”** 메시지를 표시하고 음성 안내를 실행
- 무릎 각도가 95~150도 라면 **“Lower your posture”** 메시지를 표시하고, 음성 안내를 실행

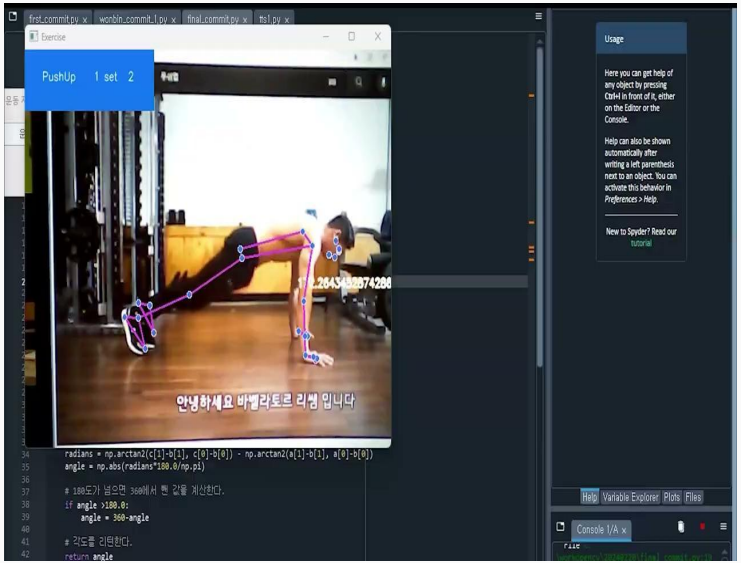
사용자가 내려간 상태 일때:

- 무릎 각도가 160도를 초과하면 다시 올라가는 동작으로 감지 및 횟수 카운트

# 04 프로젝트 결과

## 시연 영상



# 04 프로젝트 결과

## 성과 및 결과

1. 사용자의 실시간 자세 인식 및 랜드마크 표시
2. 여러 종류의 운동 자세 감지
3. 각 운동별 정자세일 때만 횟수 카운트
4. 세트 별 휴식 시간
5. 운동 자세 교정 피드백(음성, 텍스트)

# 04 프로젝트 결과

## 문제점

**문제점 :**

- **Mediapipe**가 오직 **한 사람**의 **Pose**만 추적 가능하여 영상에 여러 사람이 나올시 정확한 **Pose값 측정 불가.**
- 정확한 운동 자세 피드백이 불가능
- 사용자의 체력수준을 고려한 운동 난이도 조절불가

# 04 프로젝트 결과

## 향후 계획

1. 사용자가 운동 횟수와 세트를 **직접 선택** 할 수 있도록 조정
2. 단순한 피드백이 아니라 정확한 피드백을 할 수 있도록 구현
3. 여러 사람 혹은 **특정 개인의 Pose값** 만 추적할 수있게 조정 필요.
4. 사용자 경험 향상을 위해 추가적인 운동 종류 및 기능 개발
5. 체형인식을 통해 운동 난이도를 제안하는 기능 개발


HAVE ANY QUESTIONS??